Numerical Analysis (7/7)
Machine Learning basics

University of Luxembourg - 2024

Master in Mathematics

Hadrien Beriot

Outline

1. Introduction

What is machine learning?

- For many problems, it's difficult to program the correct behavior by hand
  - recognizing people and objects
  - understanding human speech
- Machine learning approach: program an algorithm to automatically learn from data, or from experience. Why might you want to use a learning algorithm?
  - hard to code up a solution by hand (e.g. vision, speech)
  - system needs to adapt to a changing environment (e.g. spam detection)
  - want the system to perform better than the human programmers

What is machine learning?

- **It's similar to statistics...**
  - Both fields try to uncover patterns in data
  - Both fields draw heavily on calculus, probability, and linear algebra, and share many of the same core algorithms
- **But it's not statistics!**
  - Stats is more concerned with helping scientists and policymakers draw good conclusions; ML is more concerned with building autonomous agents
  - Stats puts more emphasis on interpretability and mathematical rigor; ML puts more emphasis on predictive performance, scalability, and autonomy

Learning from data

- Learning general models from a data of particular examples
- Data is cheap and abundant (data warehouses, data marts);
- By contrast knowledge is expensive and scarce.
- Example in retail: Customer transactions to consumer behavior:

People who bought “Da Vinci Code” also bought “The Five People You Meet in Heaven”

- Build a model that is *a good and useful approximation* to the data.

Data Mining / KDD

Definition := *“Knowledge Discovery in Databases (KDD) is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data”*

Applications:

- *Retail: Market basket analysis, Customer relationship management (CRM)*
- *Finance: Credit scoring, fraud detection*
- *Manufacturing: Optimization, troubleshooting*
- *Medicine: Medical diagnosis*
- *Telecommunications: Quality of service optimization*
- *Bioinformatics: Motifs, alignment*
- *Web mining: Search engines*
- *...*

Data science lifecycle

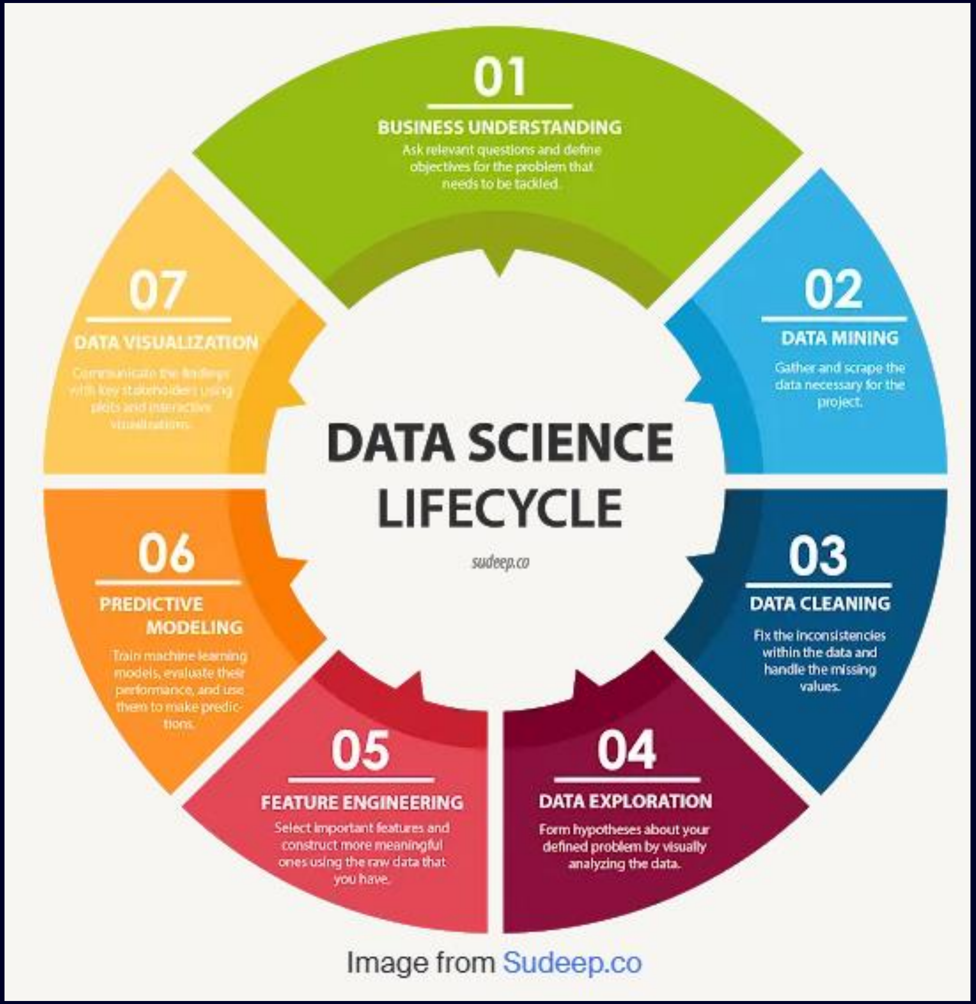
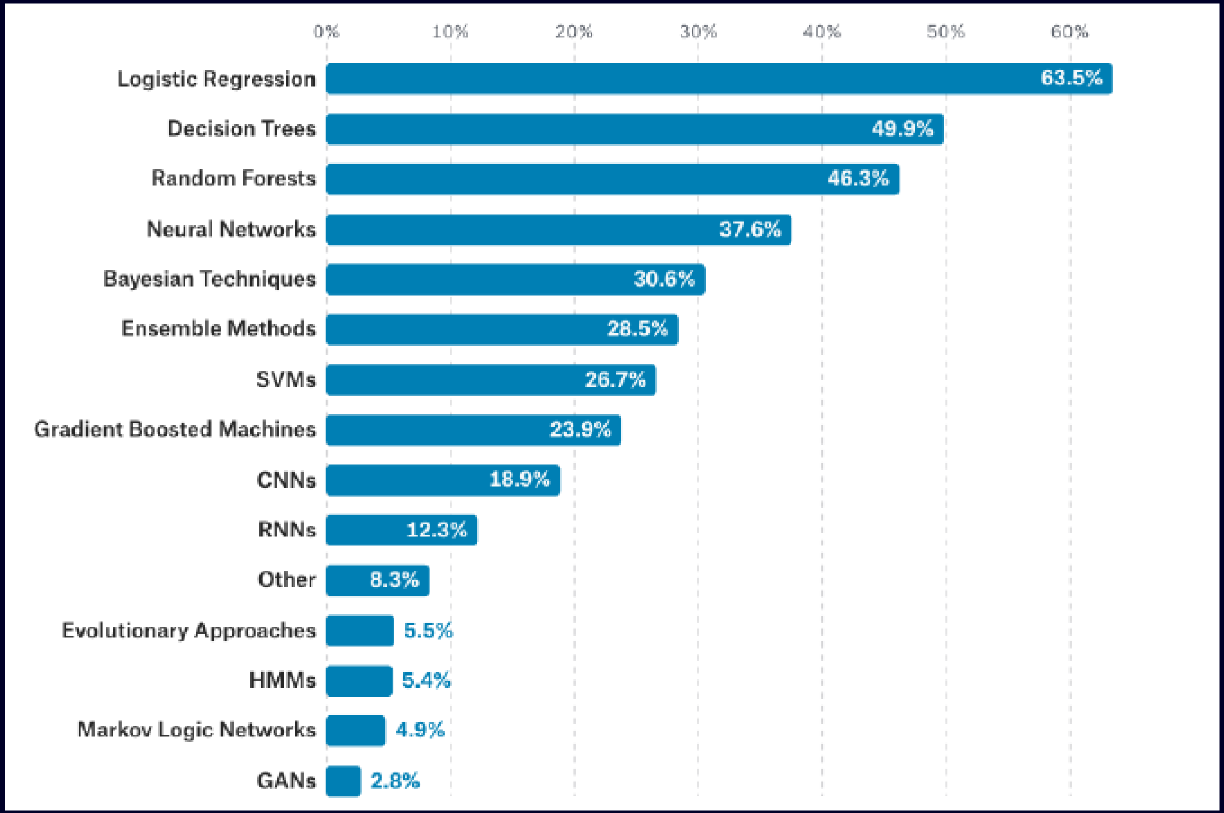

01
BUSINESS UNDERSTANDING
Ask relevant questions and define objectives for the problem that needs to be tackled.
02
DATA MINING
Gather and scrape the data necessary for the project.
03
DATA CLEANING
Fix the inconsistencies within the data and handle the missing values.
04
DATA EXPLORATION
Form hypotheses about your defined problem by visually analyzing the data.
05
FEATURE ENGINEERING
Select important features and construct more meaningful ones using the raw data that you have.
06
PREDICTIVE MODELING
Train machine learning models, evaluate their performance, and use them to make predictions.
07
DATA VISUALIZATION
Communicate the findings with key stakeholders using plots and interactive visualizations.
DATA SCIENCE LIFECYCLE
sudeep.co

Image from Sudeep.co

- It is all about the data!
- Traditional data may come from basic customer records or historical stock price information.
- **Today**: Big data mostly comes from the Web, for example, Facebook, Google, and LinkedIn; or financial trading data. This is the **web AI** (already there)
- **Tomorrow**: machine data from sensors in industrial equipment. This is **industrial AI** (still to happen)
- And the day after tomorrow: **IoT AI** (wearable tech)

Data science methods

- There are a jungle of methods
- 2017 Kaggle survey of data science and ML practitioners: what data science methods do you use at work?



Growth of machine learning

- **Machine learning is preferred approach to**
  - Speech recognition, Natural language processing
  - Computer vision
  - Medical outcomes analysis
  - Robot control
  - Computational biology
  - Language models, Translation, report generation
- **This trend is accelerating**
  - Deep learning, Large Language Models
  - Improved data capture, networking, faster computers
  - Software too complex to write by hand
  - Rise of autonomous systems, AI agents

Relations to AI

- Nowadays, “machine learning" is often brought up with “artificial intelligence" (AI)

- AI does not always imply a learning-based system
  - Symbolic reasoning
  - Tree search
  - etc.

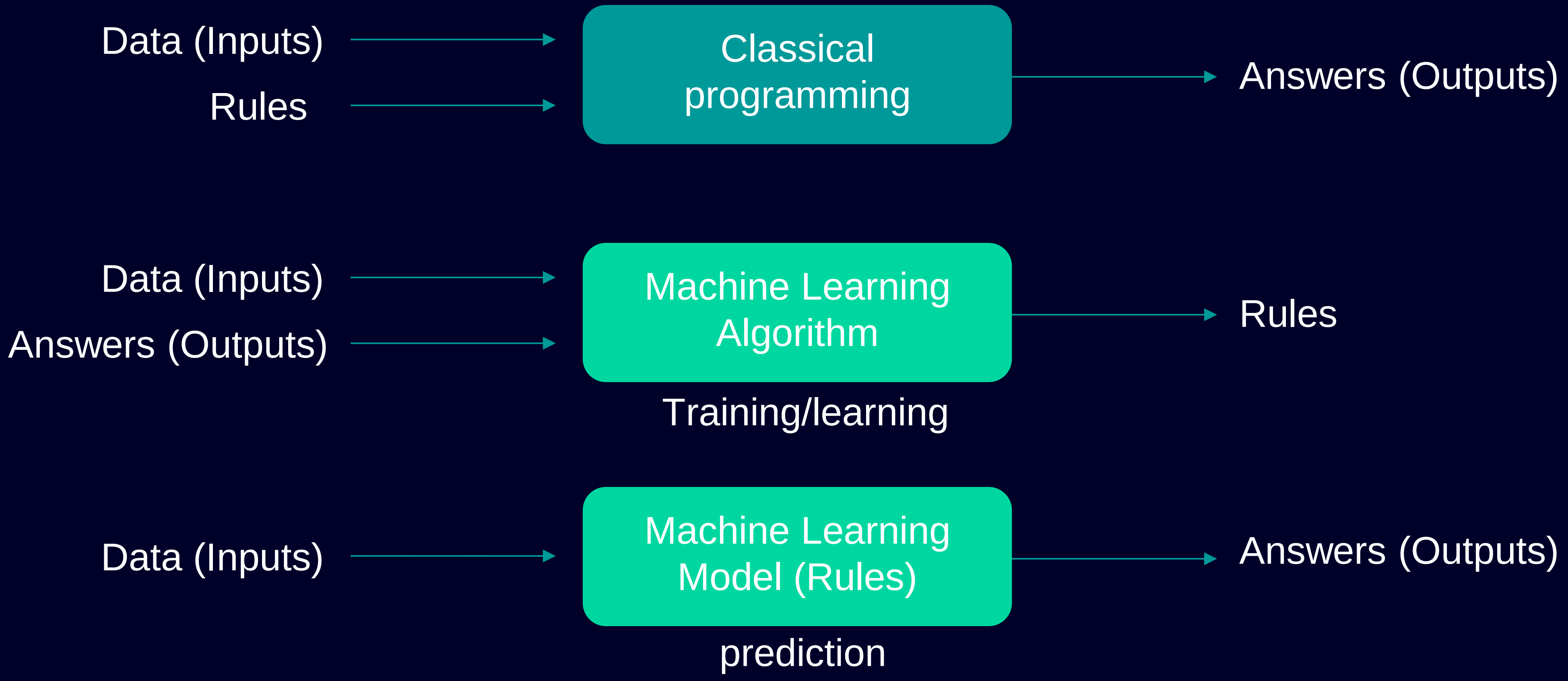
- Learning based system → learned based on the data → More flexibility, good at solving pattern recognition problems.

Relations to human learning

- Human learning is:
  - Very data efficient
  - An entire multitasking system (vision, language, motor control, etc.)
  - Takes at least a few years :)

- For serving specific purposes, machine learning doesn't have to look like human learning in the end.
- It may borrow ideas from biological systems, e.g., neural networks.
- It may perform better or worse than humans

Relations to classical programming



2. Brief history of ML

History of ML

- 1957 - Perceptron algorithm (implemented as a circuit!)
- 1959 - Arthur Samuel wrote a learning-based checkers program that could defeat him
- 1969 - Minsky and Papert's book Perceptrons (limitations of linear models)
- 1980s - Some foundational ideas
    - Connectionist psychologists explored neural models of cognition
    - 1984 | Leslie Valiant formalized the problem of learning as PAC learning
    - 1988 | Backpropagation (re-)discovered by Georey Hinton and colleagues
    - 1988 | Judea Pearl's book Probabilistic Reasoning in Intelligent Systems introduced Bayesian networks

History of ML

- 1990s - the “AI Winter", a time of pessimism and low funding
- But looking back, the '90s were also sort of a golden age for MLresearch
  - Markov chain Monte Carlo, variational inference, kernels and support vector machines
  - Boosting, convolutional networks, reinforcement learning
- 2000s - applied AI fields (vision, NLP, etc.) adopted ML
- 2010s - deep learning
  - 2010-2012 - neural nets smashed previous records in speech-to-text and object recognition
  - increasing adoption by the tech industry
  - 2016 - AlphaGo defeated the human Go champion
- 2018-now - generating photorealistic images and videos
  - 2020 - GPT3 transformer-based language model
  - 2022 – ChatGPT released to public
  - now - increasing attention to ethical and societal implications

3. Types of ML

Types of machine learning

- Types of machine learning
  - **Supervised learning**: have labeled examples of the correct behavior
    - Classification
    - Regression/Prediction
  - **Unsupervised learning**: no labeled examples - instead, looking for “interesting" patterns in the data
    - Clustering
  - **Reinforcement learning**: learning system (agent) interacts with the world and learns to maximize a scalar reward signal

Input Vectors

- Machine learning algorithms need to handle lots of types of data:
  - images, text, audio waveforms, credit card transactions, etc.
- Common strategy: represent the input as **an input vector in $\boldsymbol{R^d}$**
- Representation = mapping to another space that's easy to manipulate
- Vectors are a great representation since we can do linear algebra!

Classification (supervised learning)

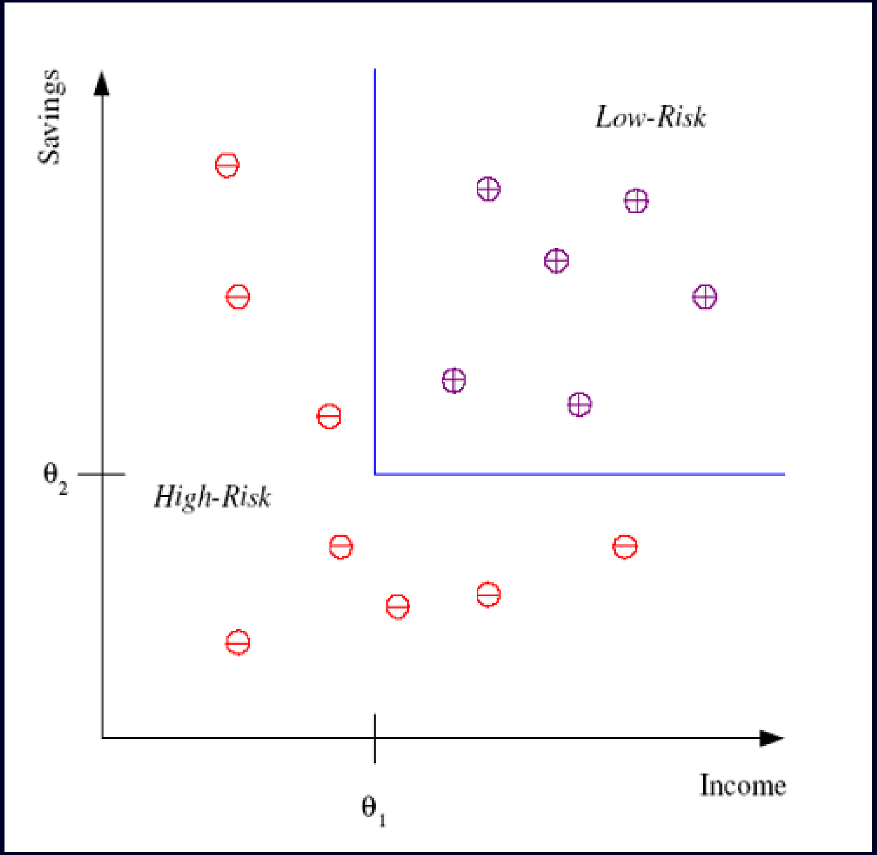
Example: Credit scoring

- Differentiating between low-risk and high-risk customers from their *income* and *savings*

- **Discriminant**: IF income > θ1 AND savings > θ2
  - THEN low-risk, ELSE high-risk

- The Discriminant here is the ***model***

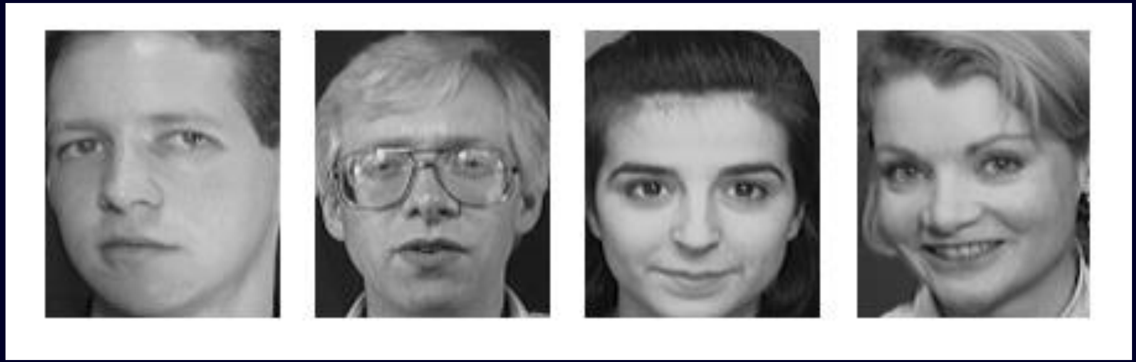
- In practice many more parameters (age, profession, number of children, …) can come in



Classification (supervised learning)

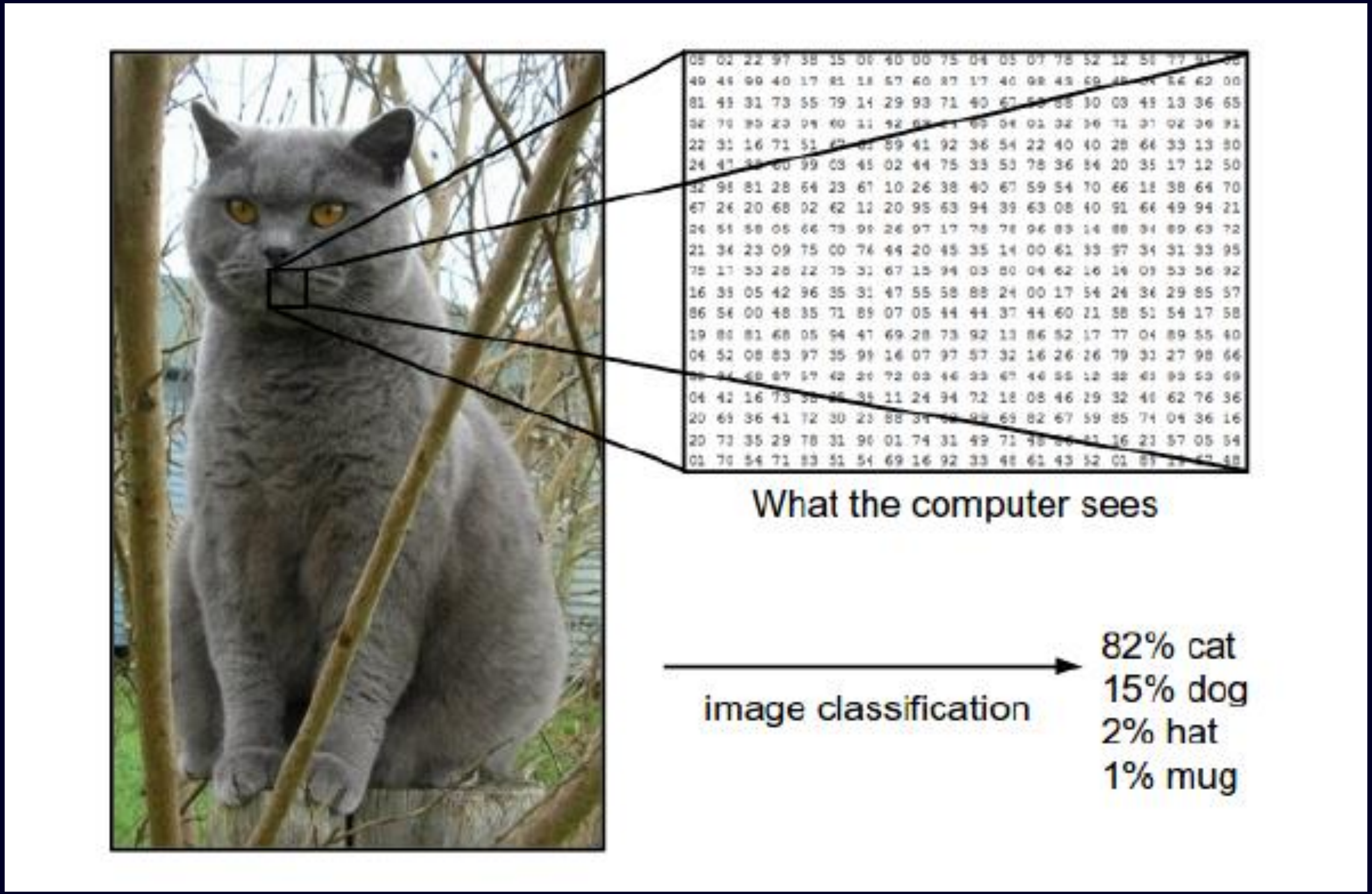
- **Face recognition**: Pose, lighting, occlusion (glasses, beard), make-up, hair style
- **Character recognition**: Different handwriting styles.
- **Speech recognition**: Temporal dependency.
- **Sensor fusion**: Combine multiple modalities; eg, visual (lip image) and acoustic for speech
- **Medical diagnosis**: From symptoms to illnesses

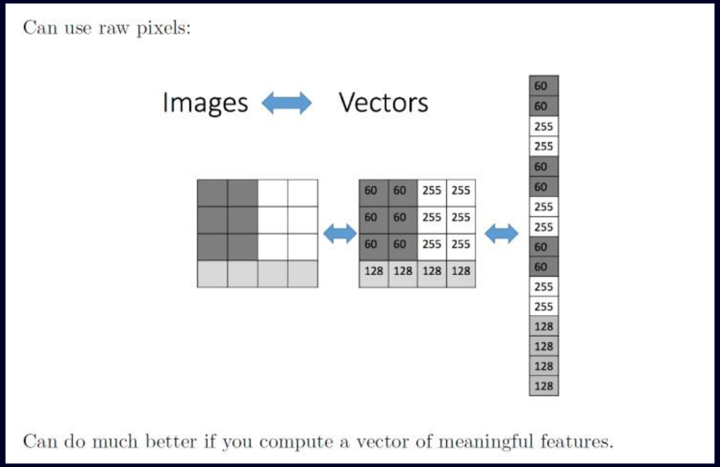
AT&T Laboratories, Cambridge UK
http://www.uk.research.att.com/facedatabase.html


Test images

Object recognition (supervised learning)

What an image looks like to the computer:

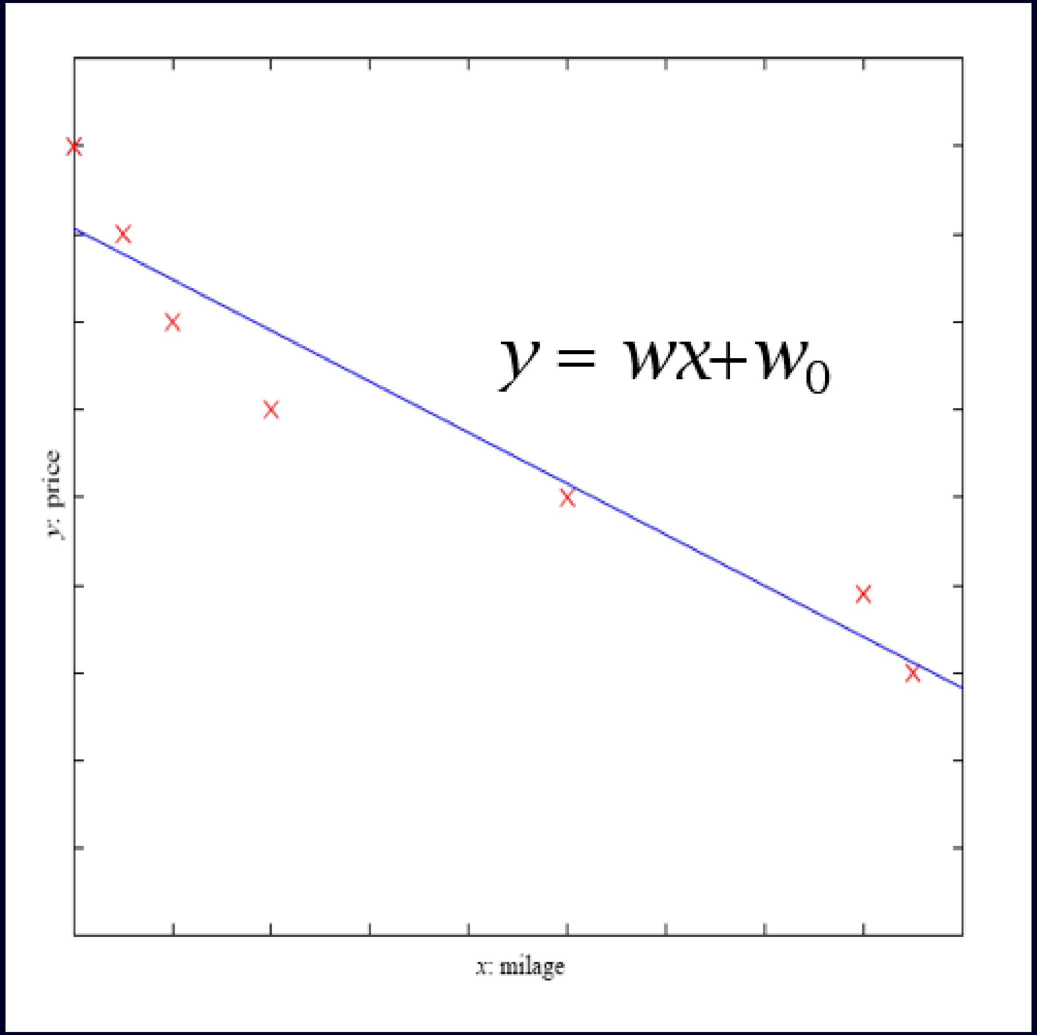



Can use raw pixels:

Can do much better if you compute a vector of meaningful features.

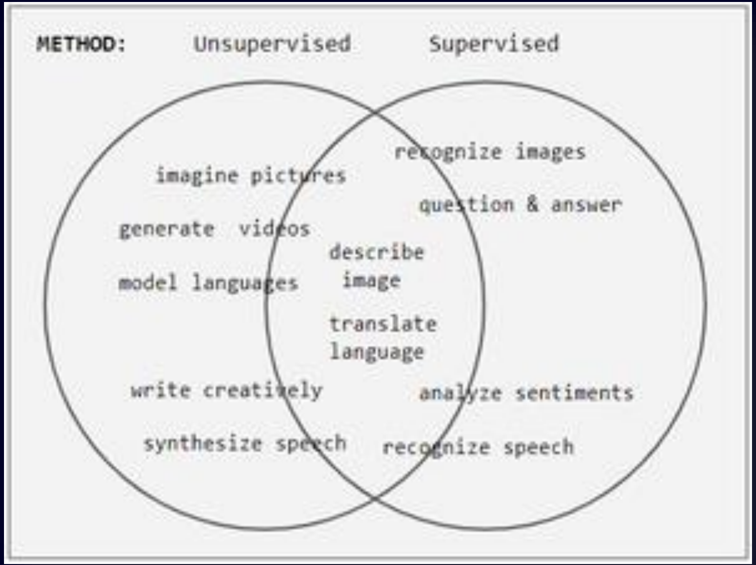
Regression (supervised learning)

- **Example**: Price of a second-hand car
  - x : car attributes
  - y : price
  - $y = g(x \mid \theta)$
  - $g(\ )$ model,
  - θ parameters



Separation not so clear

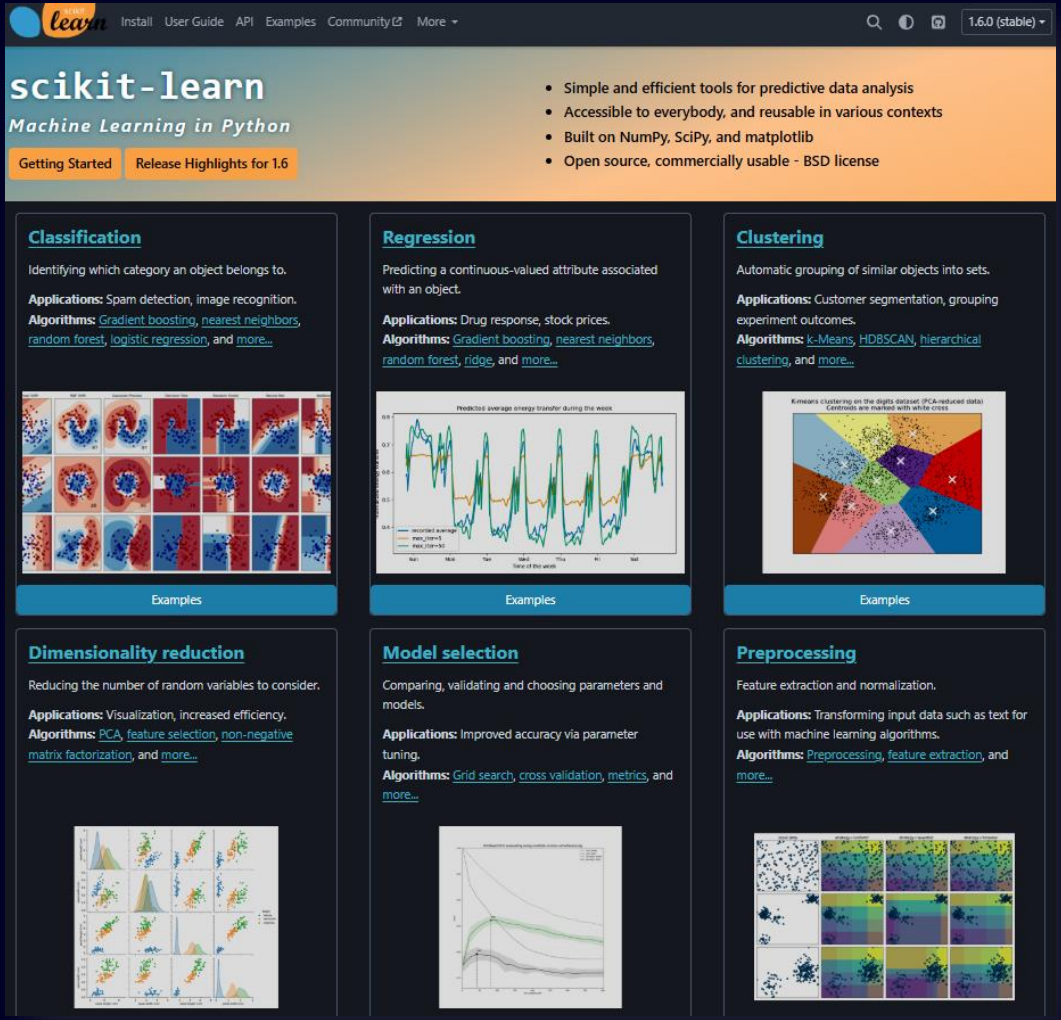
- Tasks are often categorized as discriminative (recognition) or generative (imagination).
- Often but not always, **discriminative tasks use supervised methods** and **generative tasks use unsupervised**
- However, the separation is not always clear
- For example, object recognition favors supervised learning but unsupervised learning can also cluster objects into groups



Venn diagram of ML

Scikit learn (all the basics)

- **The toolbox for python**
  - general-purpose ML learning library, covering a wide range of algorithms and techniques: classification, regression, clustering, dimensionality reduction, and more.
  - Scikit-learn is designed to be user-friendly and easy to use, with a consistent API across different algorithms.
  - It is primarily focused on classical, **non-deep learning models**, such as linear regression, decision trees, random forests, …
  - https://scikit-learn.org/stable/



scikit-learn

Machine Learning in Python

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: Gradient boosting, nearest neighbors, random forest, logistic regression, and more...

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, stock prices.
Algorithms: Gradient boosting, nearest neighbors, random forest, ridge, and more...

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, grouping experiment outcomes.
Algorithms: k-Means, HDBSCAN, hierarchical clustering, and more...

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, increased efficiency.
Algorithms: PCA, feature selection, non-negative matrix factorization, and more...

Model selection

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter tuning.
Algorithms: Grid search, cross validation, metrics, and more...

Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.
Algorithms: Preprocessing, feature extraction, and more...

Unsupervised learning

- Learning “what normally happens”
- No output
- Clustering: Grouping similar instances
- Other applications: Summarization, Association Analysis
- **Example applications**
    - natural language processing
    - image and video analysis
    - anomaly detection
    - customer segmentation
    - recommendation engines
    - …

4. Classification – nearest neighbor

Input vectors

- Mathematically, our training set consists of a collection of pairs of an **input vector** $\boldsymbol{x} \in R^d$ and its corresponding target, or label, t
  - **Regression**: t is a real number (e.g. stock price)
  - **Classification**: t is an element of a discrete set $\{1, \cdots, C\}$
  - These days, t is often a highly structured object (e.g. image)
- Denote **the training set** $\left\{\left(\boldsymbol{x}^{(1)}, t^{(1)}\right), \cdots, (\boldsymbol{x}^{(N)}, t^{(N)})\right\}$

e.g. – classification - Nearest neighbors

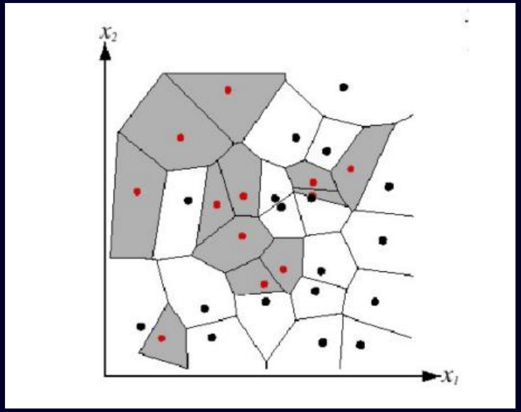
- Suppose we are given a novel input vector $\boldsymbol{x}$ we'd like to classify
  - **The idea**: find the nearest input vector to $\boldsymbol{x}$ in the training set and copy its label
  - Can formalize "nearest" in terms of Euclidean distance

$$\left|\left|\boldsymbol{x}^{(\boldsymbol{a})} - \boldsymbol{x}^{(\boldsymbol{b})}\right|\right|_2 = \sqrt{\Sigma_{j=1}^{d}\left(\boldsymbol{x}_j^{(\boldsymbol{a})} - \boldsymbol{x}_j^{(\boldsymbol{b})}\right)^2}$$

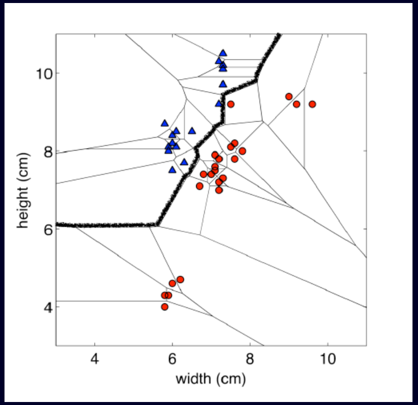
- **Algorithm:** find $(\boldsymbol{x}^*, t^*)$ (from the stored training set) closest to $\boldsymbol{x}$**.** That is:

$$\boldsymbol{x}^* = argmin_{x^{(i)} \in train.set}\ distance(\boldsymbol{x}^{(i)}, \boldsymbol{x})$$

  - Output: $y = t^*$



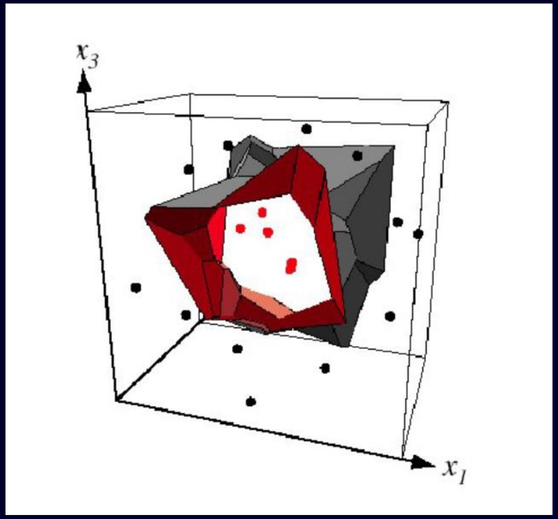
We can visualize classification setting using a Voronoi diagram



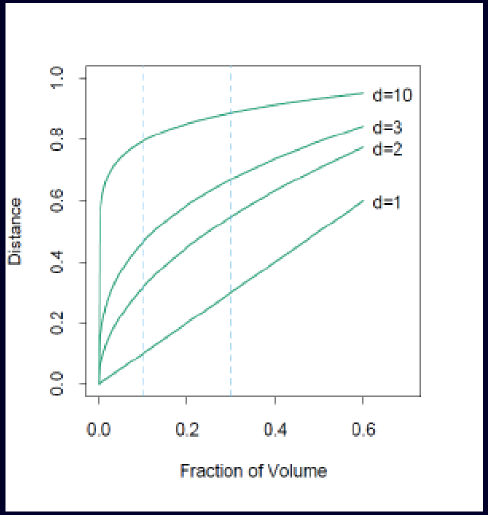
Decision boundary

e.g. – Classification - Nearest neighbors

- Low dimensional visualizations are misleading!
- In high dimensions, “most” points are far apart
- If we want the nearest neighbor to be closer than ϵ, how many points do we need?
- The volume of a single ball of radius ϵ is $O\left(\epsilon^d\right)$
- The total volume of $[0,1]^d$ is 1
- Therefore $O\left(\left(\frac{1}{\epsilon}\right)^d\right)$ balls are needed to cover the volume!
- This is the **curse of dimensionality**



Decision boundary in R^3



Curse of dimensionality

Intrinsic dimension is what matters!

- Saving grace: some datasets (e.g. images may have low intrinsic dimension, i.e. lie on or near a low-dimensional manifold

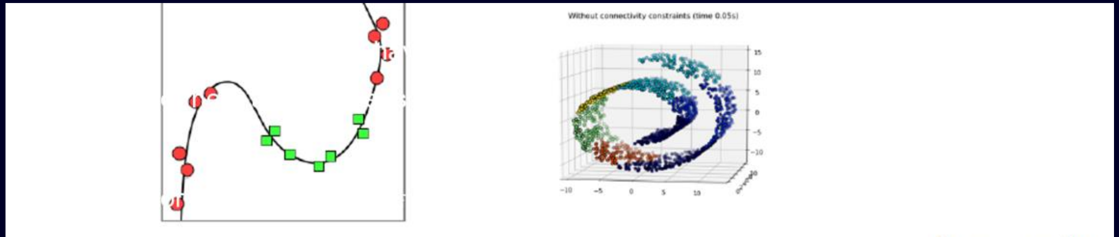
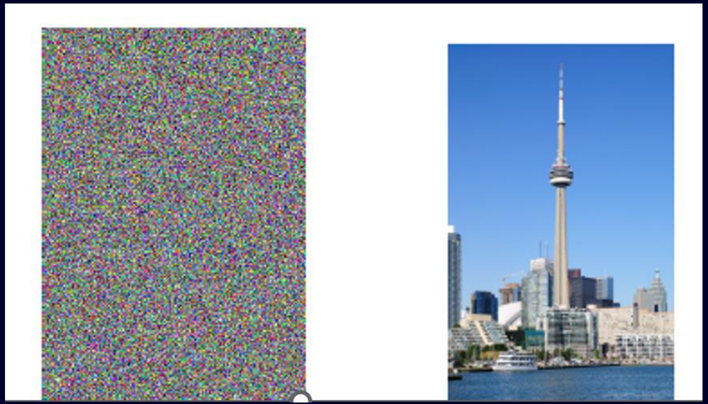


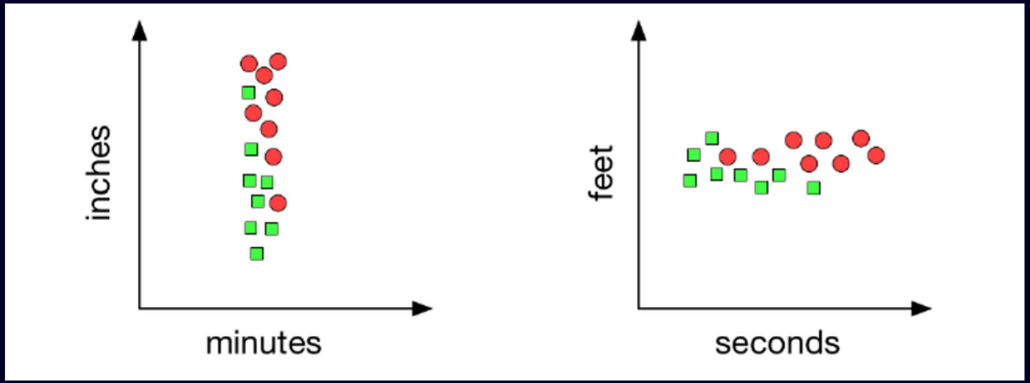
Image credit:
https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_swiss_roll.html

- The neighborhood structure and hence the curse of dimensionality depends on the intrinsic dimension
- The space of pixel images below is 3-million dimensional. The intrinsic space dimension (degrees of freedom) is much smaller (there are structures/patterns)



Normalization

- Nearest neighbors are sensitive to the ranges/scales of the different features
- Datasets units are often arbitrary

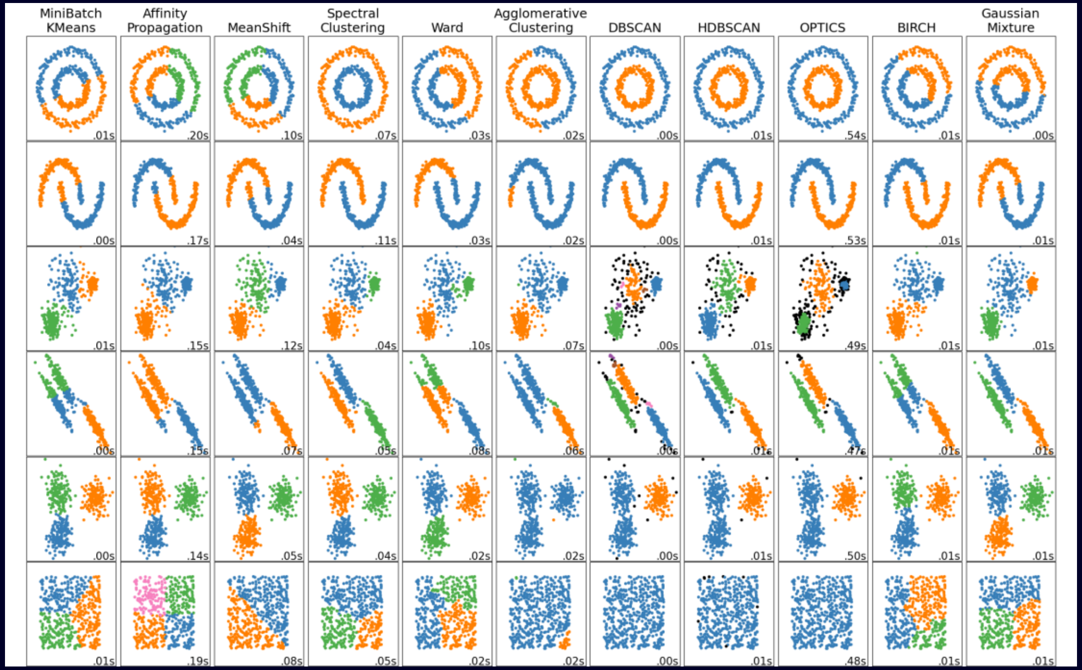


- Simple fix: **normalize** each dimension to have **zero mean** and **unit variance**
- i.e. compute the mean μ_j and variance σ_j and take

$$\tilde{x}_j = \frac{x_j - \mu_j}{\sigma_j}$$

Comparing different clustering algorithms on toy datasets

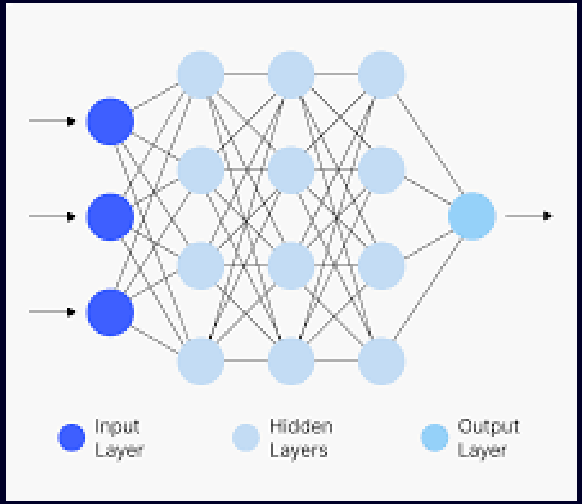
- From scikit-learn: This example shows characteristics of different clustering algorithms on datasets that are “interesting” but still in 2D. With the exception of the last dataset, the parameters of each of these dataset-algorithm pairs has been tuned to produce good clustering results.
- The last dataset is an example of a ‘null’ situation for clustering: the data is homogeneous, and there is no good clustering.
- While these examples give some intuition about the algorithms, this intuition might not apply to very high dimensional data.



https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html

5. Neural networks – multilayer perceptron

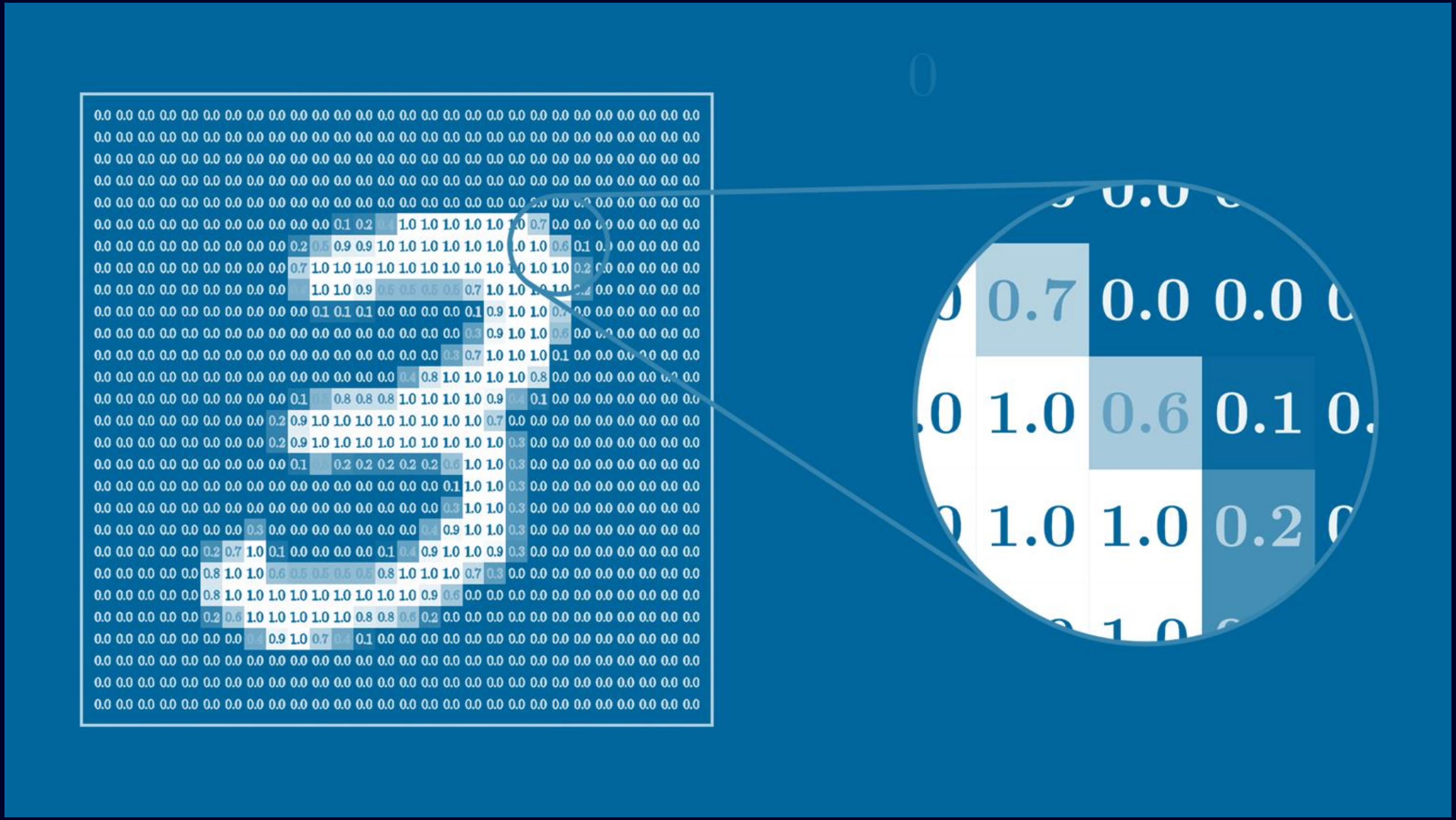
A brief overview



- **What is a Neural Network?** A neural network is a computational model inspired by the human brain. It consists of layers of interconnected nodes, called neurons, designed to recognize patterns and make predictions.

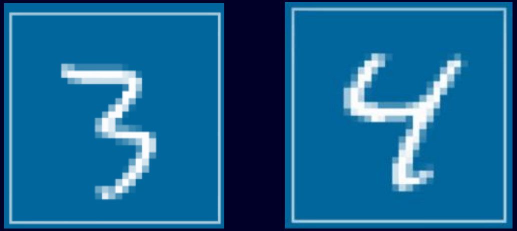
- **Neurons (Nodes):**
  - The basic units of computation in a neural network.
  - Organized into layers: input, hidden, and output.
- **Weights:**
  - Each connection between neurons has a weight.
  - Weights determine the strength of the influence of one neuron on another.
- **Bias:**
  - An additional parameter added to the input to adjust the output.
  - Helps the network model more complex relationships.
- **Activation Function:**
  - Applies a non-linear transformation to the input of a neuron.
  - Common functions:
    - Sigmoid: Maps input to a range (0, 1).
    - ReLU (Rectified Linear Unit): Outputs max(0, x)

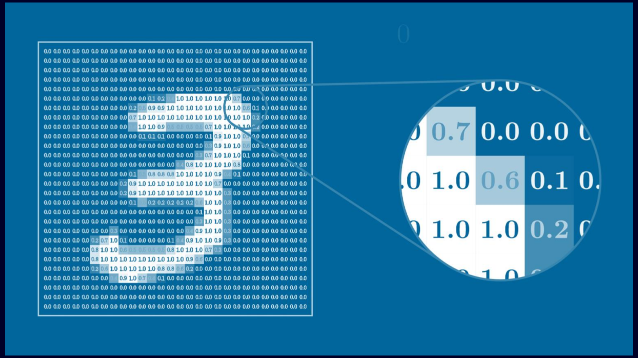
Example: Handwritten digit recognition

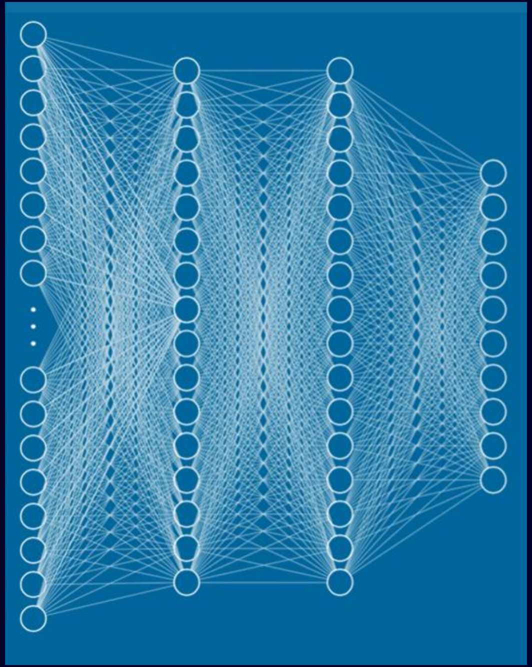


From https://www.3blue1brown.com/lessons/neural-networks

Example: Handwritten digit recognition







- The “Hello World” of Neural Networks
- https://www.3blue1brown.com/lessons/neural-networks
- 28x28 pixels with values in [0,1] (grayscale value)
- Each neuron is “lit up” when close to 1

- **Consider the following structure (*Multilayer perceptron*):**
  - Input layer = 784 neurons (one for each pixel)
  - Output layer = 10 neurons (0,1,2,3,4,5,6,7,8,9)
  - 2 hidden layers of 16 neurons each (arbitrary)

- **Total # of parameters to train:**
  - Each connection has a weight and each neuron has a bias
  - 784*16 + 16*16 + 16*10 + 16 + 16 + 10 = 13 002
- The output layer with largest number gives the **estimated digit**

Multilayer perceptrons

- Multilayer perceptrons form the basis of deep learning
- https://en.wikipedia.org/wiki/Multilayer_perceptron
- Consider the input layer with values $\boldsymbol{x} = (x_1, x_2, \cdots, x_{784})$
- Now consider the connections to the 2nd layer, we put the weights into the matrix $\boldsymbol{W}$ and the biases into the vector B

$$W = \begin{pmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{m1} & \cdots & w_{mn} \end{pmatrix} \qquad B = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$$

- Where $n = 784$ and $m = 16$. The output on second layer is obtained by

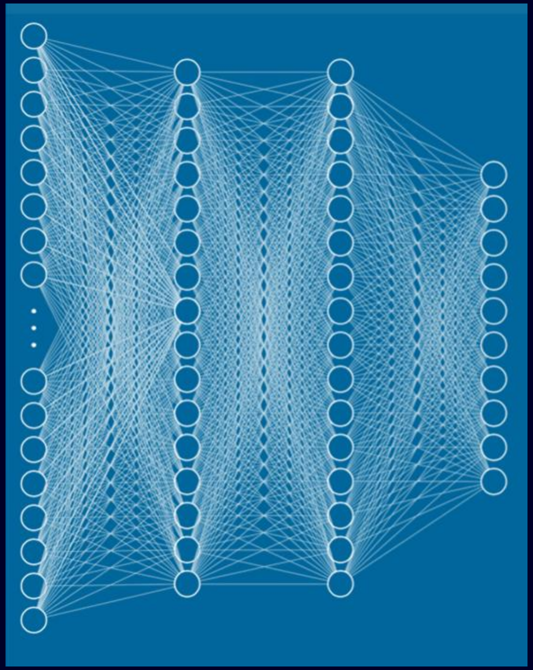
$$\boldsymbol{y} = \sigma(\boldsymbol{W}\,\boldsymbol{x} + \boldsymbol{B})$$

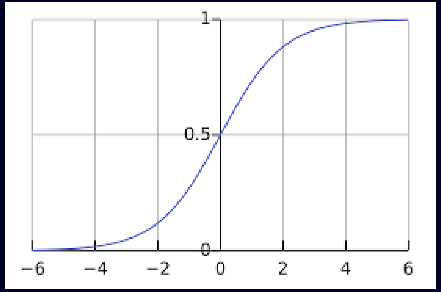
- Where σ is an activation function (**sigmoid** or ReLU). The value in the j-th neuron of the second layer is

$$y_i = \sigma(\boldsymbol{\Sigma}_{j=1}^{n}\, \omega_{ij}\, x_j + b_i)$$

- The error is measured using mean squared error (MSE)

$$MSE \;= \frac{1}{n} \boldsymbol{\Sigma}_{i=1}^{n} (\boldsymbol{y}_i - \boldsymbol{e}_i)^2$$

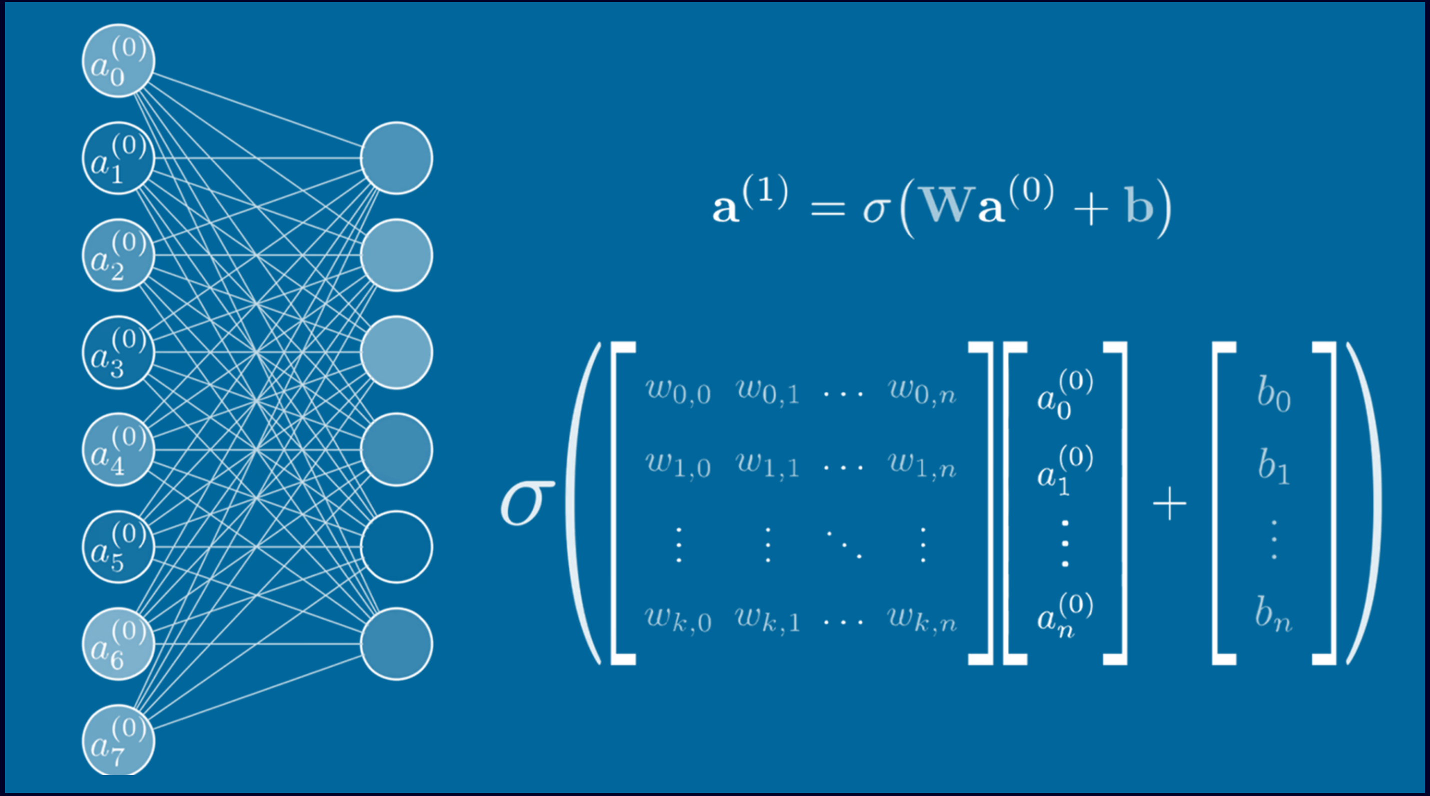

4-layer perceptron



Sigmoid activation function

Next layer activation

- **Feed forward algorithm**


$a_0^{(0)}$
$a_1^{(0)}$
$a_2^{(0)}$
$a_3^{(0)}$
$a_4^{(0)}$
$a_5^{(0)}$
$a_6^{(0)}$
$a_7^{(0)}$
$\mathbf{a}^{(1)} = \sigma(\mathbf{W}\mathbf{a}^{(0)} + \mathbf{b})$
$\sigma\left(\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \cdots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}\right)$

From https://www.3blue1brown.com/lessons/neural-networks

- **T**ransition of activations from one layer to the other $\boldsymbol{a}^{(i+1)} = \sigma(\boldsymbol{W}_i\, \boldsymbol{a}^{(i)} + \boldsymbol{b}_i)$

Learning

- You train the network with images of handwritten digits, along with labels for what they represent
- Network will adjust those 13 002 weights and biases to improve performance on **training data**
- After you trained, you test with more data not shown for training
- It comes down to **finding the minimum of a certain function**
  - Reminder: Weights = strengths of the connections, Bias = Neuron is active or inactive
- We start by assigning weights and biases randomly
- Then we estimate **the cost function, the error**:
  - Add up the squares of the differences between the proposed output layer values and the ones you want them to have = cost of a single training example

Neural Network function

- **Input:** 784 numbers (pixels)
- **Output:** 10 numbers (result)
- **Parameters:** 13002 (weights and bias)

Cost function

- **Input:** 13002 weights and biases
- **Output:** 1 number (the cost)
- **Parameters:** 10 000’s

Gradient descent

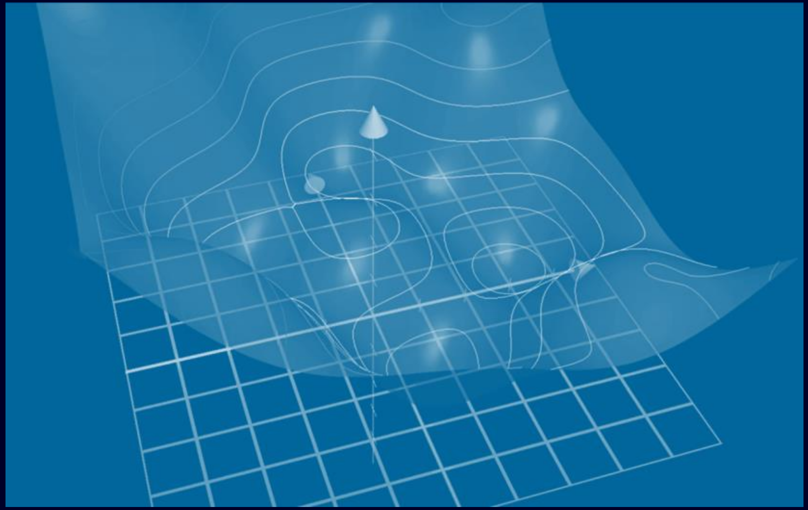
- The function we want to minimize, has 13002 dimensions!
- We can approach **a local minimum** of the function using **gradient descent**
- We go downhill the hypersurface of dimension 13002!
- What is important is to ensure the smoothness of the cost function → this is why we introduce a **smoothing activation function**

- **Neural Network function**

- **Input:** 784 numbers (pixels)
- **Output:** 10 numbers (result)
- **Parameters:** 13002 (weights and bias)

- **Cost function**

- **Input:** 13002 weights and biases
- **Output:** 1 number (the cost)
- **Parameters:** 10 000’s



Training data

- https://en.wikipedia.org/wiki/MNIST_database

- 60,000 training images and 10,000 testing images
- Labeled (exact label is available)
- Error rates down to <0.1% ("human performance“)

Back propagation

- So how do we compute the gradient?
- How do we find the delta to all the weights and biases to most efficiently decrease the cost function?
- The algorithm to compute the gradient of the cost function is called **back propagation**
- **Consider the simple network:**

$$y = \sigma(w_1 x_1 + w_2 x_2 + b) \qquad \frac{\partial C}{\partial y} = 2(y-e)$$

cost function: $C = (y-e)^2$

Gradient vector: $\nabla C = \left(\frac{\partial C}{\partial w_1}, \frac{\partial C}{\partial w_2}, \frac{\partial C}{\partial b}\right)$

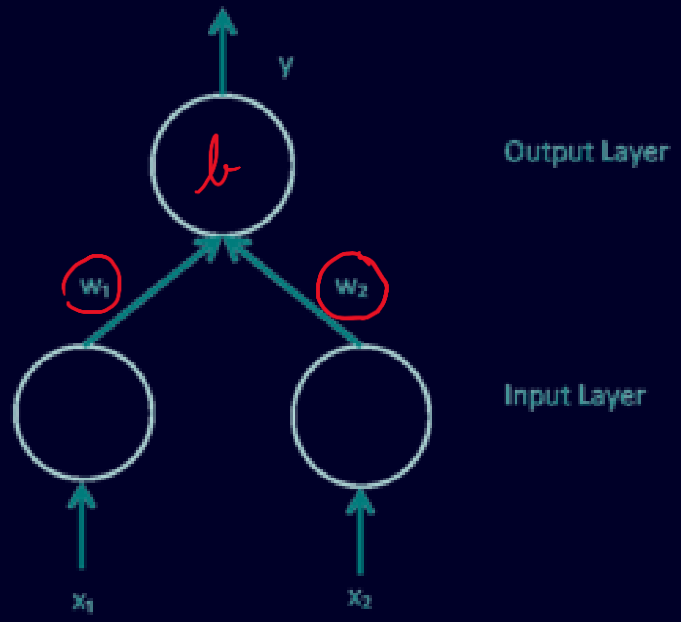
$$\frac{\partial C}{\partial w_1} = \frac{\partial C}{\partial y}\frac{\partial y}{\partial w_1} = 2(y-e)\,\underline{x_1}\,\underline{\sigma'(w_1 x_1 + w_2 x_2 + b)}$$

$$\frac{\partial C}{\partial w_2} = 2(y-e)\,x_2\,\sigma'(w_1 x_1 + w_2 x_2 + b)$$

$$\frac{\partial C}{\partial b} = 2(y-e)\,\sigma'(w_1 x_1 + w_2 x_2 + b)$$

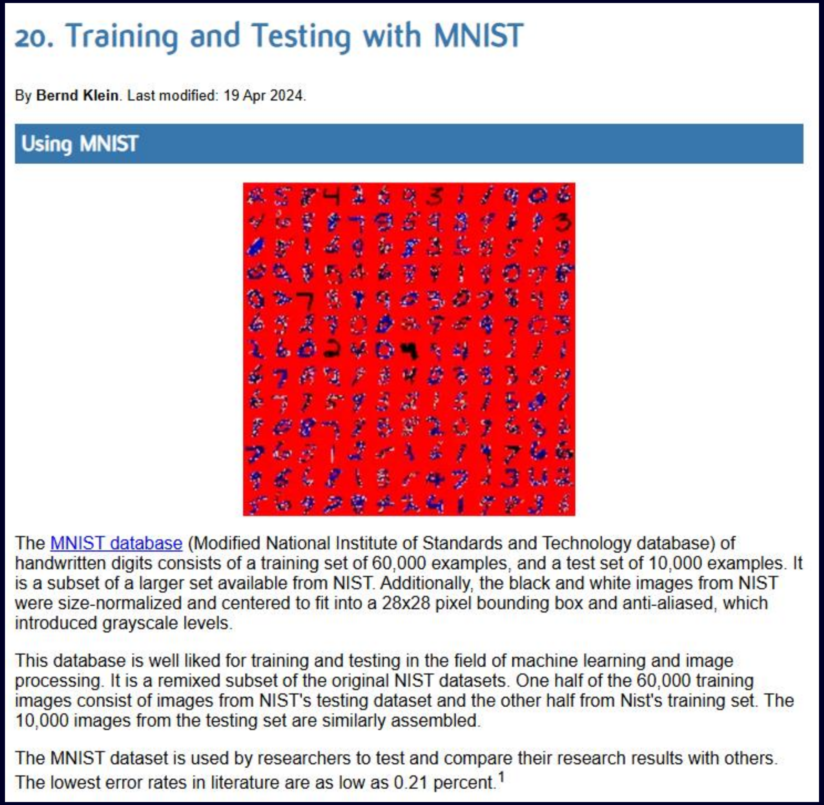
$$(g(f(x)))' = \underline{g'(f(x))}\,\underline{f'(x)}$$

$$\sigma = \frac{1}{1+e^{-x}} \qquad \sigma' = \frac{e^{-x}}{(1+e^{-x})^2}$$



MNIST example

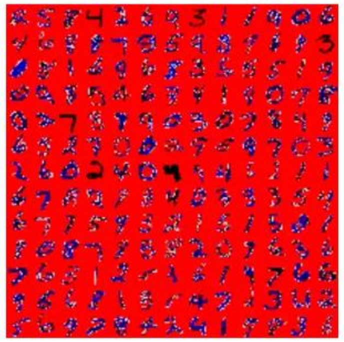
https://python-course.eu/machine-learning/training-and-testing-with-mnist.php



20. Training and Testing with MNIST

By **Bernd Klein**. Last modified: 19 Apr 2024.

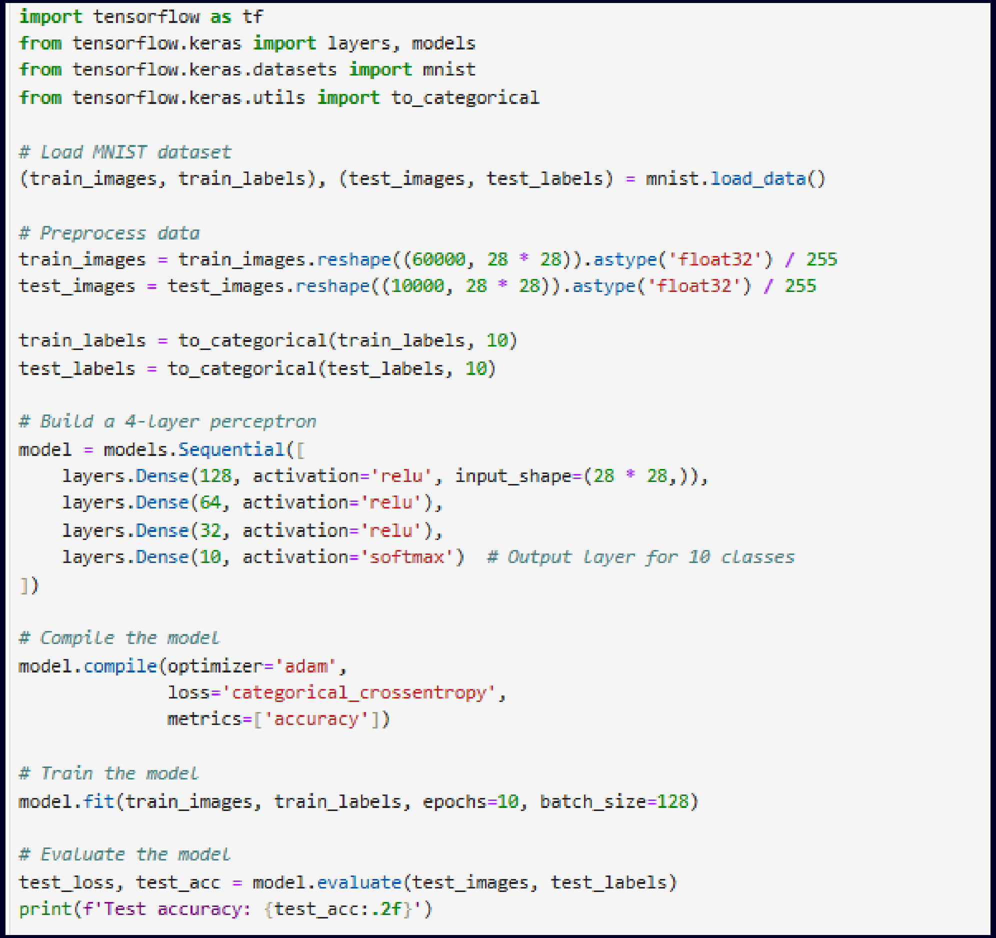
Using MNIST



The MNIST database (Modified National Institute of Standards and Technology database) of handwritten digits consists of a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. Additionally, the black and white images from NIST were size-normalized and centered to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels.

This database is well liked for training and testing in the field of machine learning and image processing. It is a remixed subset of the original NIST datasets. One half of the 60,000 training images consist of images from NIST's testing dataset and the other half from Nist's training set. The 10,000 images from the testing set are similarly assembled.

The MNIST dataset is used by researchers to test and compare their research results with others. The lowest error rates in literature are as low as 0.21 percent.[1]



```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

# Load MNIST dataset
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

# Preprocess data
train_images = train_images.reshape((60000, 28 * 28)).astype('float32') / 255
test_images = test_images.reshape((10000, 28 * 28)).astype('float32') / 255

train_labels = to_categorical(train_labels, 10)
test_labels = to_categorical(test_labels, 10)

# Build a 4-layer perceptron
model = models.Sequential([
    layers.Dense(128, activation='relu', input_shape=(28 * 28,)),
    layers.Dense(64, activation='relu'),
    layers.Dense(32, activation='relu'),
    layers.Dense(10, activation='softmax')  # Output layer for 10 classes
])

# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
model.fit(train_images, train_labels, epochs=10, batch_size=128)

# Evaluate the model
test_loss, test_acc = model.evaluate(test_images, test_labels)
print(f'Test accuracy: {test_acc:.2f}')
```

6. Summary

Summary

- Machine learning is becoming a classical tool for engineers, and comes in complement to traditional programming, when rules or logic cannot be defined a-priori
- It requires vast amount of data, and consists in two steps, training and inference
- Supervised training requires labeling the data a-priori, unsupervised training finds associations
- For classification, nearest neighbor is based on minimizing Euclidean distance in large dimensional spaces. While spaces are of high dimension, data can often be represented in a lower dim. space
- For more complex tasks (e.g. shape recognition), Neural Networks, like Multilayer Perceptrons may be needed.
  - It requires a structure of layered neurons with activation functions and biases, fed forward through weighted connections
  - A cost function, including all weights and biases is minimized during Back-propagation using a simple **gradient descent**